Taylor Polynomial Approximations of a Function

Note: You may notice differences between this Maple worksheet and the equivalent Mathematica notebook. These differences were introduced to preserve the content of these modules and were necessary because of major functional differences between Maple and Mathematica.

Introduction

OBJECTIVE: Observe an animated demonstration of the convergence of Taylor polynomials to a function that has derivatives of all orders over some interval of its domain.

In this module we write a short procedure that can be used for demonstrations and explorations. The loop demonstrates the convergence of Taylor polynomials to a function over some interval of its domain, provided the function has derivatives of all orders. The command generates a plot that can be animated or viewed frame by frame, to demonstrate the convergence of the Taylor polynomials. Several functions are included in the module, and others can easily be added. Part I leads you through the construction of the functions used in this worksheet and explains how it works.

Technology Guidelines

NOTE: If you have just finished a worksheet, **restart** *Maple* before executing a new worksheet.
TO OPEN SECTIONS,
Click on the **PLUS** sign at the left hand side of the screen *or* select **Expand All Sections** from the **View** drop down menu.

TO STOP AN EXECUTION
Click on **STOP** button from the toolbar.

ORDER OF EXECUTION
Execute commands in the order given. Do not skip any *Maple* Input lines within a given worksheet

Alternatively, you can execute the entire worksheet by selecting the **Execute Worksheet** command from the **Edit** drop down menu.

SAVING WORKSHEETS.

You can save anytime to any directory you choose, and it is wise to save often.
EXPERIENCING MAJOR PROBLEMS
Save if appropriate, and then shut down *Maple* and start it up again.

Part I: Computing and Visualizing Taylor Polynomials

First, we load the **plots** package.

```
> restart:
  with(plots):
```

Warning, the name changecoords has been redefined

We can use *Maple* to find Taylor polynomial approximations for functions in the vicinity of a point whose *x*-coordinate equals *a*. The third-degree polynomial that approximates **exp(x)** near $x = 0$ is:

```
> unassign('x'):
  series(exp(x),x=0,4);
```

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + O(x^4)$$

The last term is called the error term. It tells us that the cubic polynomial gives a fourth-order approximation of **exp(x)** near $x = 0$.

To evaluate a Taylor polynomial for specific values of *x*, we must obtain a form of the polynomial that does not include the error term. This can be done as follows.

```
> convert(series(exp(x),x=0,6),polynom);
```

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5$$

Next, we build our own *Maple* command that will give an n^{th} degree Taylor polynomial for approximating **f** near $x=a$.

```
> taylor2:=proc(f,a,deg):
    convert(series(f,x=a,deg+1),polynom):
  end;
```

$$taylor2 := \mathbf{proc}(f, a, deg)\ \mathrm{convert}(\mathrm{series}(f, x = a, deg + 1), polynom)\ \mathbf{end\ proc}$$

The next command shows how **taylor2( )** works.

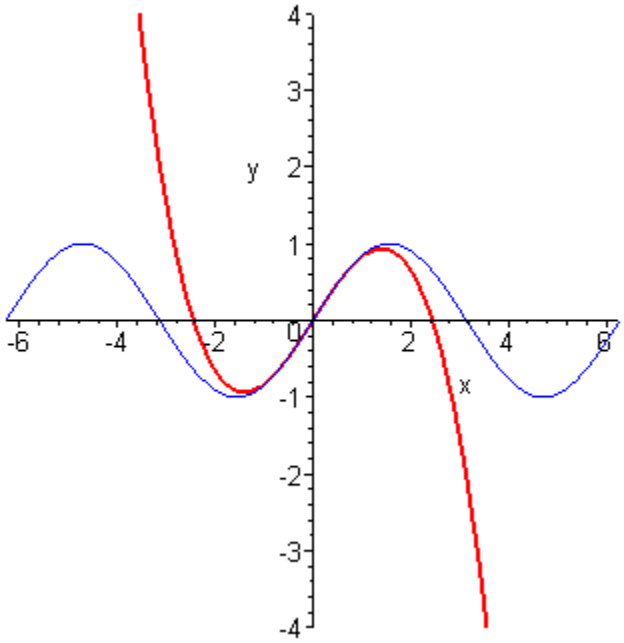
```
> taylor2(exp(x),0,7);
```

$$1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4+\frac{1}{120}x^5+\frac{1}{720}x^6+\frac{1}{5040}x^7$$

Now we use our new **taylor2( )** command to demonstrate how Taylor polynomials converge on the **sin(x)** as **deg**, the degree of the polynomial, increases. Execute the commands in the next cell, then repeatedly increase the degree of the Taylor polynomial (**deg**) and re-execute the commands.

```
> unassign('f,g,x'):
  f:=sin(x):
  a:=0:
  deg:=3:
  g:=taylor2(f,a,deg):
  graph1:=plot(f(x),x=-2*Pi..2*Pi, y=-4..4, color=COLOR(RGB,0,0,1), labels=[x,y]):
  graph2:=plot(g(x),x=-2*Pi..2*Pi, thickness=2, color=COLOR(RGB,1,0,0), labels=[x,y]):
  print(`The third degree polynomial to approximate Sin(x) is`, taylor2(f,0,a));
  display(graph1,graph2);
  print(`The function is plotted in blue: it's Taylor polynomial is in red`);
```

The third degree polynomial to approximate Sin(x) is, 0



The function is plotted in blue: it's Taylor polynomial is in red

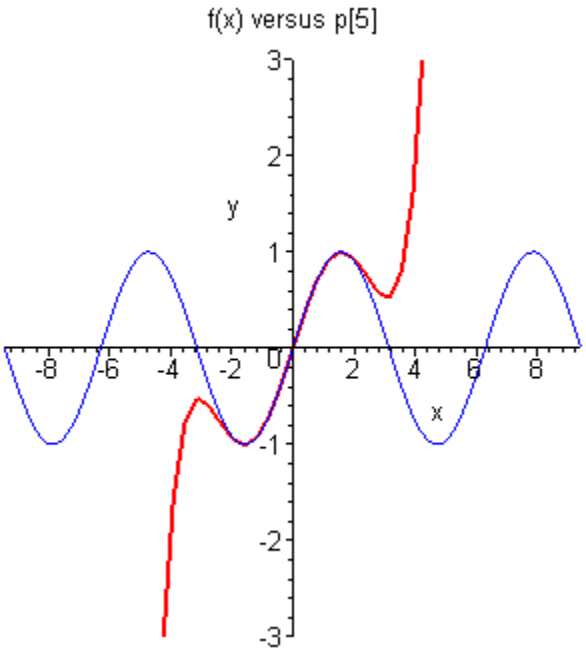
We can put this all together and form a new command, which we call **compare( )**.

```
> unassign('f,g,x,deg'):
  compare:=proc(f,a,deg,xlimits, ylimits)
    local g,graph1,graph2:
    g:=taylor2(f,a,deg):
    graph1:=plot(f(x),x=xlimits, y=ylimits, color=COLOR(RGB,0,0,1),      labels=
```

```
[x,y],discont=true):
  graph2:=plot(g(x),x=xlimits, thickness=2,color=COLOR(RGB,1,0,0), labels=[x,y]):
  display(graph1,graph2,title=`f(x) versus p[`||deg||`]`);
end:
```

Let's try it on $\sin(x)$ near x=0

```
> compare(sin(x), 0, 5, -3*Pi..3*Pi, -3..3);
```



You Try It: Part I

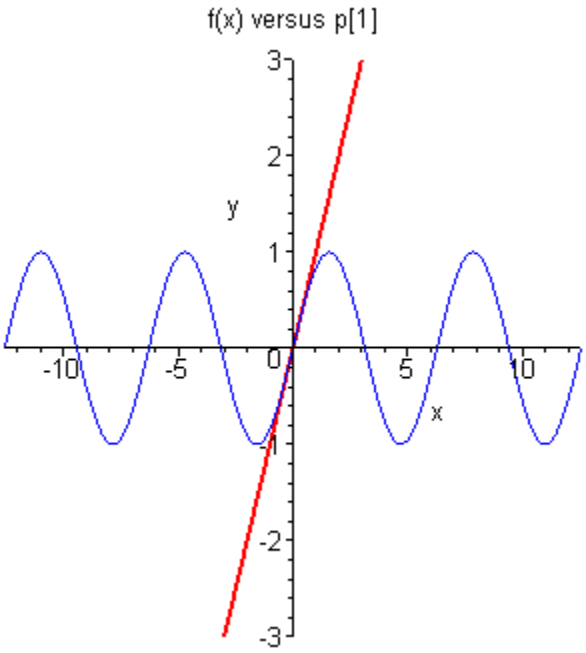
Try some of your own functions. Some suggestions are: **cos(x)** at $a = 0$, **exp(x)** at $a = 0$, **1/(1+x)** at $a = 0$, **ln(x)** (the natural log) at $a = 1$, and **sqrt(x)** at $a = 4$. Change the appropriate entries in the next cell. The commands that follow generate the **Taylor** polynomial and graph it together with the function. The example shows the 5th-degree Taylor polynomial approximation for ln x around $a = 1$. For each function you try, be sure to increase the degree of the Taylor polynomial. You can also adjust the limits on the viewing window.

```
f:=ln(x):
a:=1:
deg:=5:
xlimits:=0.001..4:
ylimits:=-4..5:
print(`The Taylor polynomial to approximate`, f, `around`, a, ` is`, taylor2(f, a, deg));
compare(f,a,deg,xlimits,ylimits);
print(`The function is plotted in blue; it's Taylor polynomial is in red.`);
```

Part II: Demonstrations of Convergence and Lack of Convergence

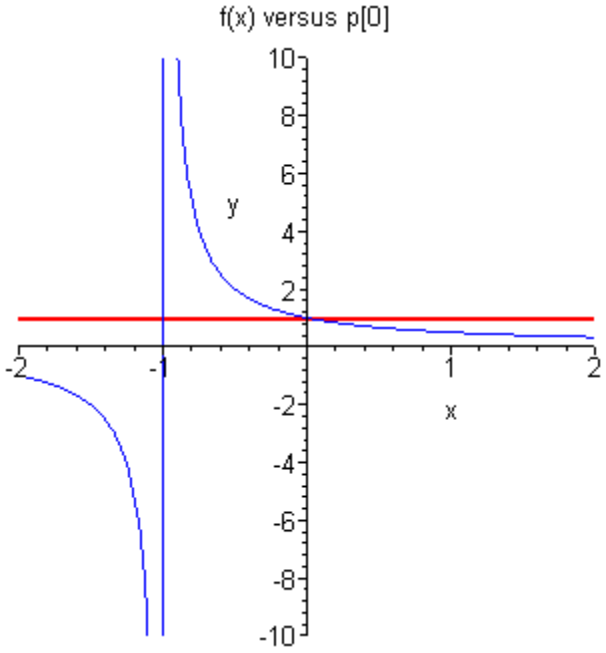
The next command generates an image that you can animate. To animate the graph, click on the plot that is generated and select the **play button** (big arrow) from the animation toolbar at the top of the worksheet.. The animation toolbar only appears after you have clicked on the animation plot. Click on either of the **double arrow buttons** to slow down or speed up the animation. Click on the **frame by frame button** (arrow pointing to a vetical bar) to view the animation frame by frame. The **frame by frame button** is recommended.

> **display(seq(compare(sin(x), 0, 2*i+1, -4*Pi..4*Pi, -3..3), i=0..12), insequence=true);**



The sine function you just plotted is very well behaved. Consider the following rational function, and see what happens. You can animate the plots in the sequence by clicking on the graph and then selecting play buttons in the animation tool bar at the top of the worksheet.

> **display(seq(compare(1/(1+x), 0, deg, -2..2, -10..10), deg=0..20), insequence=true);**

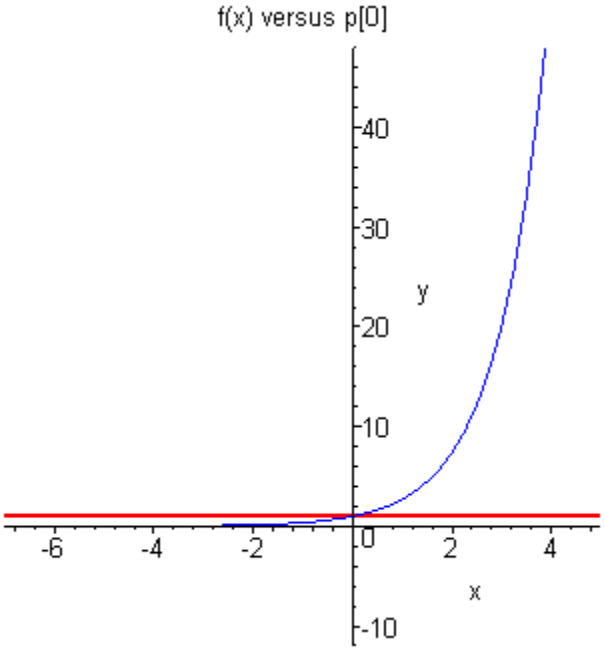


>

You Try It: Part II

Select your own function, and change the appropriate items in the following cell. Be careful to use correct *Maple* format when inserting your own functions. Some functions that you may want to try, if you haven't already, are: **cos(x)** at $a = 0$, **exp(x)** at $a = 0$, **1/(1+x)** at $a = 0$, **ln(x)** (the natural log) at $a = 1$, and **sqrt(x)** at $a = 4$. The commands in the next cell are for **exp(x)** at $a = 0$. You can change the parameters inside the compare function to test different functions, degrees, and a-values. You can also adjust the limits on the viewing window.

```
> f:=exp(x):
  a:=0:
  maxdeg:=14:
  xlimits:=-7..5:
  ylimits:=-12..48:
  display(seq(compare(f, 0, deg, xlimits, ylimits), deg=0..maxdeg),insequence=true);
```


f(x) versus p[0]
40
30
y
20
10
-6
-4
-2
0
2
4
x
-10

>